R Notebook

Gabriela Suardi

2023-03-08

In this file I'll put my notes from John Hopkins' data science course R programming

Lesson 1: R console Imput and evaluation

<- assignment operator : assign a value to a symbol "#" indicates a comment. Everything on the right of this symbol is ignored Example:

```
x<- 1     ## nothing printed
x         ## autoprinting occurs
```

```
## [1] 1
```

```
print("The value of x is")     ##explicited printing
```

```
## [1] "The value of x is"
```

```
print(x)
```

```
## [1] 1
```

The : operator is used to create interger sequences Example:

```
x <- 1:20  #this gives us a sequence
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

*** Lesson 2: R objects and attributes***

R has five basic classes of objects: **-character -numeric (real numbers) -integer - complex -logical (true/false)**

The most basic object is a *vector* - Everything in R is a object -A vector can only contain objects of the *same* class. Example: a vector containing characters and numerics.

BUT the only exception is a **list**, which is represented as a vector but can contain objects of different classes Empty vectors can be created with the *vector()* function

The vector function has two basic arguments. The **first argument** is the *class* of the object, so the type of object that you want to have in the vector. And the **second argument** is the *length* of the vector itself.

Numbers Numbers in R are generally treated as *numeric objects* If you explicit wants an integer, you need to use the suffix **L**. Ex: 1L There's also the special number **inf** which represents infinity. The **NaN** (not a number) represents an undefined value or a missing value.

Lesson 3: Creating vectors

The c() function (concatenating) can be used to create vectors of an object

```
a <- c(0.5, 0.6) ##number
a
```

```
## [1] 0.5 0.6
```

```
b <- c(TRUE, FALSE) ##logical
d <- c(T, F)   ##logical
e <- c("1", "a", "d", "0.8" ) ##character
f <- 9:29 ##integer
f
```

```
##  [1]  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
```

```
g <- c(1+0i, 2+4i) ##complex
g
```

```
## [1] 1+0i 2+4i
```

Using the vector function:

```
x <- vector("numeric", length=10)
x
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

When different objects are mixed in a vector, **coercion** occurs so every object is of the same class When converted to numeric, TRUE is represented by the number 1 and FALSE by 0.

Explicit coercion Objects can be explicit coerced from one class to another Use of the function as.

```
x <- 0:6
class(x)
```

```
## [1] "integer"
```

```
as.numeric(x)
```

```
## [1] 0 1 2 3 4 5 6
```

```
as.logical(x) #as a convention, 0 is false and every other number above zero is true
```

```
## [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
as.character(x)
```

```
## [1] "0" "1" "2" "3" "4" "5" "6"
```

Nonsensical coersions results in *NA*

```
x <- c("a", "b", "c")
as.complex(x)
```

```
## Warning: NAs introduzidos por coerção
```

```
## [1] NA NA NA
```

```
as.numeric(x)
```

```
## Warning: NAs introduzidos por coerção
```

```
## [1] NA NA NA
```

```
as.logical(x)
```

```
## [1] NA NA NA
```

Lists Lists are a special type of vector that can contain elements from different classes.

```
x <- list(1, "a", TRUE, 1+4i)
x
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] 1+4i
```

Lesson 4: Matrices

Matrices are vectors with a dimension attribute. The dimension attribute itself is an integer vector of length 2

```r
m <- matrix(1:6, nrow = 2, ncol = 3)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```r
dim(m)
```

```
## [1] 2 3
```

```r
attributes(m)
```

```
## $dim
## [1] 2 3
```

```r
dim
```

```
## function (x)  .Primitive("dim")
```

Matrix can also be created directly from vectors by introducing a dimensional attribute

```r
m <- 1:10
dim(m)<- c(2, 5) # I'm assigning an attribute to the vector m
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
```

Matrices can be created by columm binding or row binding using cbind() and rbind()

```r
x <- 1:3
y <-10:12
cbind(x, y)
```

```
##      x  y
## [1,] 1 10
## [2,] 2 11
## [3,] 3 12
```

```r
rbind(x, y)
```

```
##   [,1] [,2] [,3]
## x    1    2    3
## y   10   11   12
```